METHOD FOR NETWORK-EFFICIENT DISTRIBUTED SEARCH AND DECISION-MAKING USING CO-EVOLUTIONARY ALGORITHMS EXECUTING IN A DISTRIBUTED MULTI-AGENT ARCHITECTURE

STATEMENT AS TO RIGHTS TO INVENTIONS MADE UNDER FEDERALLY SPONSORED RESEARCH AND DEVELOPMENT

FIELD AND BACKGROUND OF THE INVENTION

[0002]   The present invention relates generally to the field of distributed decision problem solving and to problems which require information retrieved from

multiple, logically interrelated, distributed databases. More specifically, the invention relates to utilization of co-evolutionary algorithms executing in a distributed information architecture having co-evolutionary agents and mobile software agents for performing the problem solving in a network-efficient manner.

[0003]   Previous work at the Electronics Agile Manufacturing Research Institute (EAMRI) at Rensselaer Polytechnic Institute has recognized the critical role which information infrastructure plays in the design and manufacturing organizations. Research at EAMRI has resulted in the development of a virtual design module, described in U.S. Patent 6,249,714, for using evolutionary agents emanating from a central location to produce optimized designs.

[0004]   An understanding of evolutionary algorithms and distributed problem solving methods is helpful to the discussion of the invention.

[0005]   Evolutionary algorithms (EAs) include genetic algorithms, evolutionary programming, evolution strategies and genetic programming. Genetic algorithms are discussed in David E. Goldberg, "Genetic Algorithms In Search, Optimization, and Machine Learning." Addison-Wesley, Massachusetts 1989 and John H. Holland, "Adaptation in Natural and Artificial Systems: an introductory analysis with applications to biology, control, and artificial intelligence." The MIT Press, Cambridge, Massachusetts, 3d ed. 1994. L.J. Fogel et al.,

"Artificial Intelligence Through Simulated Evolution."
John Wiley, New York 1966, provide a good explanation
of evolutionary programming. More detail about
evolution strategies is found in Thomas Back,
"Evolutionary Algorithms in Theory and Practice."
Oxford University Press, New York 1996. J. Koza,
"Genetic Programming: On the programming of computers
by means of natural selection." The MIT Press,
Cambridge, Massachusetts 1992 provides information
about the title topic.

[0006]   The principles of each of these related
techniques define a general paradigm that is based on
a simulation of natural evolution. Evolutionary
algorithms perform their search by maintaining at any
time t a population: $\mathcal{P}(t) = \{P_1(t), P_2(t), P_3(t), \ldots, P_p(t)\}$ of individuals. Genetic operators
that model simplified rules of biological evolution
are applied to create a new and desirably more
superior population $\mathcal{P}(t+1)$. The genetic evolution
process continues until a sufficiently good population
is achieved, or some other termination condition is
satisfied.

[0007]   Each $P_i(t) \in \mathcal{P}(t)$ represents, via an internal data
structure, a potential solution to the original
problem. Choice of an appropriate data structure for
representing solutions is very much an art, rather
than a science, due to the plurality of data
structures suitable for a given problem. However,
choice of an appropriate representation is often a
critical step in a successful application of

evolutionary algorithms. Effort is required to select a data structure that is compact, minimally superfluous, and can avoid creation of infeasible individuals in the population.

[0008]   As an example, if the problem domain requires finding an optimal integer vector from the space defined by dissimilarly bounded integer coordinates, it is more appropriate to choose as a representation an integer-set-array instead of a representation capable of generating bit strings. An integer-set-array is an array of bounded sets of integers, while a representation that generates bit strings can create many infeasible individuals, and is certainly longer than a more compact sequence of integers.

[0009]   Closely linked to choice of representation of solutions, is the choice of a fitness function: $\psi : \mathcal{P}(\cdot) \to \mathbb{R}$, that assigns credit to candidate solution. Individuals in a population are assigned fitness values according to some evaluation criterion. Fitness values measure how well individuals represent solutions to the problem. Highly fit individuals are more likely to create offspring by recombination or mutation operations. Weak individuals are less likely to be selected for reproduction, and so they eventually die out. A mutation operator introduces genetic variations in the population by randomly modifying some of the building blocks of individuals.

[0010]   Evolutionary algorithms are essentially parallel by design, and at each evolutionary step a breadth search of increasingly optimal subregions of the

options space is performed. Evolutionary search is a powerful technique of solving problems, and is applicable to a wide variety of practical problems that are nearly intractable with other, conventional optimization techniques. Practical evolutionary search schemes do not guarantee convergence to the global optimum in a predetermined finite time, but they are often capable of finding very good and consistent approximate solutions. However, they are shown to asymptotically converge under mild conditions, as described in R. Subbu et al., "Modeling and convergence analysis of distributed coevolutionary algorithms," Proceedings of the IEEE Congress on Evolutionary Computation, San Diego, California 2000 and R. Subbu, "Network Decision Support based on Distributed Coevolutionary Algorithms," PhD thesis, Rensselaer Polytechnic Institute, Troy, New York 2000.

[0011]   Distributed coevolutionary computations are a further innovation of evolutionary algorithm problem solving. Parallel and distributed implementations of evolutionary algorithms typically follow a coarse-grained approach of evolving independent populations on multiple nodes and occasionally migrating individuals between nodes. Alternatively, the fine-grained approach can be used in which individuals are distributed among multiple nodes where they have localized interactions. These approaches are discussed in greater detail in M. Capcarrere et al., "A statistical study of a class of cellular evolutionary algorithms," Evolutionary Computation

vol. 7, issue 3, 1999.

[0012]   In each of the coarse- and fine-grained methods, each node in the system can potentially directly manipulate variables in all n dimensions.   These models have primarily been pursued for the purpose of speeding computations in large-scale problems and for simultaneously alleviating the problem of premature convergence.

[0013]   Coevolutionary algorithms are distributed and consist of distinct distributed algorithm components that considered together follow various models of cooperation or competition. In this model, different subspaces of the feasible space are explored concurrently by the algorithm components.   If a problem is such that the subproblem solved by each algorithm component is independent of the others, that is, the problem is decomposable, then each algorithm component can evolve without regard to the other components.   From the perspective of optimization analysis, in such cases, each algorithm component optimizes in a landscape disjoint from the landscapes corresponding to the other algorithm components.

[0014]   However, many problems exhibit complex interdependencies, and from a coevolutionary perspective, it has been suggested that the effect of changing one of the interdependent subcomponents leads to a deformation or warping of the landscapes associated with each of the other interdependent subcomponents. Kauffman et al. discuss this theory in "Co-evolution to the edge of chaos: Coupled fitness

landscapes, poised states, and co-evolutionary avalanches," Artificial Life II, 1990. Recent studies have increased the interest in application of coevolutionary systems to problem solving.

[0015]   Husbands et al., "Experiments with an ecosystems model for integrated production planning" Handbook of Evolutionary Computation, Edited by Back et al., Oxford University Press 1997, propose a coevolutionary distributed genetic algorithm for integrated manufacturing planning and scheduling. In this scheme, each species concentrates on identifying a feasible set of process plans for a single component to be manufactured. The species interact because they utilize shared manufacturing resources. The individual fitness functions of each species take into account the need to utilize shared resources, and are based on various manufacturing costs. In order to resolve conflicts between species, an arbitrator species is simultaneously evolved. The fitness of the arbitrator species depends on its ability to resolve conflicts such that manufacturing delays are minimized. The individuals in each species compete internally in order to generate good process plans, and species compete at a higher level for shared manufacturing resources.

[0016]   Others, especially Potter et al., "Cooperative Coevolution: an Architecture for Evolving Coadapted Subcomponents," Evolutionary Computation, vol. 8, issue 1, 2000, have proposed a coevolutionary model in which multiple species evolve independently, enter

into temporary collaborations with certain members of the other species, and are rewarded based on the success of the collaboration in solving a problem. A collaboration of all species is required to realize a coherent and complete problem solution. In this model, typically the best individual from each species is chosen as the representative that will collaborate with individuals of the other species. Thus, for evaluating the fitness of each individual in a given species, the best representatives from each of the other species are utilized to form the complete solution, following which the solution is evaluated. The fitness is assigned strictly to the individual being evaluated and is not shared with the representatives from the other species that participated in the collaboration. This method which utilizes only the best individual from each population for fostering across-species collaborations could be characterized as "greedy", due to the lack of sharing between representatives. The greedy method is effective in its way because of its simplicity. However, this simple pattern of interaction between species leads to entrapments in local optima. Other collaborations schemes that include random selections of representatives are also possible, and lead to improved results over the "greedy" method.

[0017]   A competitive coevolutionary model applicable to scheduling problems is proposed by F. Seredynski, "Competitive coevolutionary multi-agent systems: The application to mapping and scheduling problems,"

Journal of Parallel and Distributed Computing, vol. 47, issue 1, 1997. Seredynski considers game-theoretic models of limited interaction between individuals in competing populations. Individuals in a population have limited interaction with individuals in the neighboring populations, and seek to maximize their fitness based on local evaluations. Seredynski demonstrates the successful emergence of global behaviors achieved purely through local cooperation.

[0018]   Coevolutionary approaches that follow a clearly adversarial model are based on the biological belief that an adaptive change of a species introduces a new challenge to the competing species. The challenge of the first change then causes a second adaptive change by the competing species, which in turn causes a response in the first species and so on. See, C.D. Rosin, "Coevolutionary Search Among Adversaries," PhD thesis, Univ. of Calif. San Diego, San Diego, California 1997. In these adversarial systems, the fitness of an individual in a population is based on a competition with members from the other population. Rosin has applied the adversarial coevolutionary model to various problems, including for instance, to the design of drugs that are robust across some drug resistance mutations, and to game playing.

[0019]   Distributed problem solving (DPS) is another problem solving method in which a cooperative solution to a problem is generated by loosely coupled agents operating according to a decentralized computational model. In this case, loosely coupled means that the

agents spend the majority of their time computing and working the problem rather than communicating with other agents. The distributed problem solving model does not have a centralized data store, and no one agent has enough information to make a complete decision. Each agent in the system requires assistance from at least one other agent in the decision-making process. Further, agents may be physically and logically distributed over a computing environment. The problem is solved by intelligently combining subproblems solved by agents into an overall solution.

[0020] The fundamental areas of interest in distributed problem solving are the decomposition and coordination of computation among a society of agents so that structural demands of the task domain are matched. See, B. Chandrasekaran, "Natural and social system metaphors for distributed problem solving: Introduction to the issue," IEEE Transactions on Systems, Man, and Cybernetics, SMC-11(1), 1981.

[0021] One of the earliest projects in distributed problem solving is the Contract-Net-Protocol, described in Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," IEEE Transactions on Computers, C-29(12), 1980 and Smith et al., "Frameworks for cooperation in distributed problem solving," IEEE Transactions on Systems, Man, and Cybernetics, SMC-11(1), 1981. In Contract-Net-Protocol, computing nodes coordinate their activities through contracts.

A "manager" node announces a task for which multiple eligible "contractor" nodes respond with bids. Contractor nodes receive pieces of the contract after a negotiation process, and in case a contractor requires assistance with its part of the problem, it assumes the role of a manager and subcontracts its part of the problem to other nodes. The original problem is solved in a top-down manner by a network of contractor nodes. The method for task decomposition is specified a priori, and the Contract-Net framework is best suited to problems that can be hierarchially decomposed into nearly independent subtasks.

[0022]   Others have proposed DPS systems which work effectively despite inconsistencies. See, Lesser et al., "Functionally accurate, cooperative distributed systems," IEEE Transactions on Systems, Man, and Cybernetics, SMC-11(1), 1981. In the "functionally accurate, cooperative" approach proposed by Lesser et al., nodes cooperatively exchange and integrate partial and tentative results to construct a complete solution. Nodes make progress in problem solving by using whatever information they can find. This work is motivated by the argument that consistency maintenance at all times is very expensive in practical systems. This model, however, leads to the possibility that agents might propagate and use incorrect partial results leading to unpredictable system performance.

[0023]   In Lesser, "A retrospective view of FA/C distributed problem solving," IEEE Transactions on

Systems, Man, and Cybernetics, vol. 21, issue 6, 1991, the issue of functionally accurate cooperative distributed problem solving is revisited. Lesser presents some techniques that may reduce the unpredictability of systems that propagate and use incorrect partial results. Lesser proposes an increase in sophistication of local control in each agent so available information about a local search is more efficiently utilized, the exchange of meta-level information between agents so their local searches can be made while having a more global view, and satisfying control, in which less than optimal, but acceptable, levels of coordination between agents are used.

[0024]   Another approach provides an architecture for solving distributed search problems using heuristics and constraint satisfaction methods, and applies it to decentralized job-shop scheduling. See, Sycara et al., "Distributed constrained heuristic search," IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, issue 6, 1991.

[0025]   One application in particular for distributed problem solving is the design and fabrication of new products. Only ten or fewer years ago, the designing, testing and manufacturing of products took place within the homogeneous environments of relatively large companies. Now, more often than not, the fabricating, assembling and testing of new products happens in widely distributed and radically different settings. The product designer, component suppliers,

and manufacturers are often many miles apart, and sometimes separated by countries and continents. The computing systems used by each group are often different, using different CAD/CAM systems and different computer platforms from PC's to workstations to communicate with each other. Despite the best efforts of all involved, there are usually delays in filling orders, the end-products are more costly, less reliable and take longer to make than is possible.

[0026]   Although advanced network backbones and technologies for communications exist, new product design and fabrication has in some ways seemed to remain a manual process. Many times an optimal product may be designed by engineers who then transmit their design to suppliers for cost estimates and manufacturing reviews, which reveal that the design is far from optimal in cost and production time.

[0027]   Some e-engineering systems are beginning to improve the design and realization processes. Some systems rely upon collaborative design processes by providing shared design files and collaborative design development among distributed engineering participants. Other systems provide for rapid communications and file transfers for design assessment and checking among distributed design contributors including potential customers as well as fabrication services and other supply chain contractors. Still other systems merely provide access to databases with time-critical data needed by design engineers.

[0028] Unfortunately, each of these prior art approaches are limited solutions to the inherently complex and coupled problem of making ideal design, supplier and manufacturing choices with respect to multiple criteria including cost and time. Fundamental to this complexity is that each choice or assignment has the potential to affect overall product cost, time to manufacture and distribute, and therefore assignments cannot be considered independent of one another. This complexity makes it increasingly impossible to make these selections based purely on prior art techniques.

[0029] The Virtual Design Environment class of systems automates significant portions of the overall decision task by accessing information available at multiple, logically interrelated, distributed databases and evaluating and consequently optimizing the assignments. These systems rely heavily on the underlying network infrastructure and middle-ware to radically alter the design development and realization process.

[0030] The Virtual Design Environment (VDE) described in U.S. Patent 6,249,714 and developed in part by the inventors of this invention is a distributed, heterogeneous information architecture having centralized evolutionary computations. The VDE uses evolutionary agents, modular computer programs that generate and execute queries among distributed computing and database resources and support a global optimization of integrated design-supplier-manufacturer planning decisions.

[0031]   A prototype VDE has been evaluated using design and supplier data based on a real commercial electronic circuit board product for Pitney Bowes of Stamford, Connecticut, and data from three commercial manufacturing facilities.   The VDE simultaneously selects parts for a design, selects suppliers for the parts and makes manufacturing decisions.   Suppliers and manufacturing resources are distributed, and information about parts, suppliers and manufacturing resources is available through network databases. During the course of the evolutionary optimization, the VDE generates virtual designs, or complete integrated planning decisions, that are evaluated against an evaluation function based on cost and time models.   These computations require information collected dynamically over the network.   As the evolution proceeds, successive generations of virtual designs are created, and the population systematically converges towards promising integrated planning decisions.

[0032]   The VDE functions in a network environment with distributed information sources that need to be accessed for decision-making.   In this network environment, inter-node communication delays are a primary factor in system performance. The prototype VDE is conceptually based on a centralized optimization model, where computation is performed only at one node, while information resident at various network nodes is frequently accessed in order to perform fitness evaluations of the alternative

planning decisions. Such a centralized planner that requires information from multiple network-distributed databases is prone to large delays due to remote access latencies.

[0033] Thus, there is a need to provide an effective problem solving system which overcomes these delays while providing the functionality of the VDE.

SUMMARY OF THE INVENTION

[0034] It is an object of the present invention to provide a new method and architecture for solving complex problems using a class of efficient and scalable algorithms in a distributed network environment.

[0035] It is a further object of the invention to provide a method and system for solving problems using coevolutionary algorithms in a distributed network having multiple nodes in which a solution to a given problem can be obtained from any node in the system.

[0036] Accordingly, a class of efficient and scalable algorithms incorporating rules of natural selection is adapted to a networked architecture for performing a distributed network-based search and consequential decision making. The algorithms are applied at each node in a logical cluster of nodes. In alternating manner, the algorithms are applied using information available in local databases connected to each node, followed by transferring information between nodes using mobile agents. An optimized solution is

produced from available information quickly and efficiently by evolutionary agents performing local operations at their respective nodes and swapping progressively increasing optimized information between evolutionary agents at different nodes using mobile agents.

[0037]   In accordance with a method of the invention, a distributed network architecture has one or more logical clusters containing nodes. Coevolutionary agents are programmed with evolutionary algorithms at each node relevant to the problem being solved throughout the architecture. The coevolutionary agents perform local searches of available information, following which they swap information between nodes according to one of six schemes, using mobile agents to transmit information. The local searching and information swapping are repeated to eventually produce an optimized result which can be obtained from any node in the architecture actively working on the problem.

[0038]   The various features of novelty which characterize the invention are pointed out with particularity in the claims annexed to and forming a part of this disclosure. For a better understanding of the invention, its operating advantages and specific objects attained by its uses, reference is made to the accompanying drawings and descriptive matter in which a preferred embodiment of the invention is illustrated.

BRIEF DESCRIPTION OF THE DRAWINGS

[0039]   In the drawings:

[0040]   Fig. 1   is a schematic diagram of a distributed network architecture supporting the problem solving method of the invention;

[0041]   Fig. 2   is a diagram showing the relationship between virtual design problem elements; and

[0042]   Fig. 3   is a diagram of a network for solving design problems using the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0043]   In centralized mode of operation, such as described in U.S. Patent 6,249,714, a network distributed search and design application using evolutionary agents has one node where an evolutionary agent is resident. The remaining nodes in the network participate in the search by simply providing information to the evolutionary agent upon request. In this mode, a search of the full space of the system takes place from only the one node occupied by the evolutionary agent, while the remaining nodes simply respond to queries from the agent. Based on the responses received, the evolutionary agent creates and evaluates virtual designs, and uses proportional selection and stochastic variational operations to evolve virtual designs for evaluation.

[0044]   The present invention, by contrast, provides a solution method and architecture in which multiple

evolutionary agents operating at different, distributed nodes all work to solve the same problem simultaneously.

[0045] Referring now to the drawings, in which like reference numerals are used to refer to the same or similar elements, Fig. 1 illustrates a distributed network architecture 10 for supporting multiple coevolutionary agents 30a, 30b, 30c, 30d spread among several nodes 20a, 20b, 20c, 20d. Each node 20a-20d includes a networked computer 25a-25d, a connected local database 50a-50d, an evolutionary agent 30a-30d and several mobile agents 60.

[0046] Each of the nodes 20a-20d shown in Fig. 1 may be a member of a logical cluster of nodes networked together in a local network, as will be further described herein. Further, while only four nodes 20a-20d are illustrated, there may be as few as 2 nodes and up to any number of nodes which can actively work together on the same network.

[0047] The evolutionary agents 30a-30d are actually co-evolutionary agents because they can evolve simultaneously with each other, using some overlapping information and some unique information. Each evolutionary agent 30a-30d includes primary search variables 32a-32d and secondary search variables 34a-34d. The search variables 32a-32d and 34a-34d are partitioned among the evolutionary agents 30a-30d. The evolutionary agent 30a-30d at each of the nodes 20a-20d performs a local evolutionary search using its corresponding primary search variable 32a-32d. The

local evolutionary search is based on local and rapidly accessible information from the corresponding local database 50a-50d. During the local evolutionary search, the secondary variables 34a-34d are clamped, or held constant.

[0048]   Following execution of the local evolutionary search, the secondary variables 34a-34d at each node 20a-20d are updated by intercommunication between the nodes 20a-20d. Mobile agents 60 are used to effect the intercommunication between the nodes 20a-20d by carrying information from an originating node to a destination node. The mobile agents 60 provide missing computational functionality at the nodes 20a-20d where they migrate.

[0049]   The local search phase and intercommunication phases are alternated to produce a cooperative search by the nodes 20a-20d, guided by the same objective search function.

[0050]   The evolutionary agent 30a-30d at each node 20a-20d performs the following functions. Each evolutionary agent 30a-30d implements a local evolutionary algorithm that searches over the subspace corresponding to locally available information in the local database 50a-50d. Each evolutionary agent 30a-30d initializes using appropriate information that permits the agent 30a-30d to do local decision-making. The evolutionary agents 30a-30d each generate and execute queries on the corresponding local database 50a-50d. Finally, the evolutionary agents 30a-30d co-exist in a pool of evolutionary agents, and

participate in coordinating a global computation of a given problem via interactions with other ones of the evolutionary agents 30a-30d and mobile agents 60.

[0051]   The coordination of the evolutionary agents 30a-30d is most critical, since a coordination operation essentially provides an updated view of the local information from a certain node 20a-20d to another of the nodes 20a-20d where that information is not currently available locally. That is, the coordination function permits the several evolutionary agents 30a-30d to co-extensively evolve based on their local searches, while being fed new information from other nodes 20a-20d between searches.

[0052]   When more than one node exists in a logical cluster of nodes 20a-20d, the virtual designs generated by each node 20a-20d in the logical cluster compete with each other during the coordination operation. This function allows local solutions generated by the evolutionary agents 30a-30d at each of the nodes within a logical cluster to compete against all of the other local solutions produced. Further, the subproblems solved by each node 20a-20d in a logical cluster are different, despite being functionally similar. That is, the subproblems are different because of the differences in local resources, such as local databases 50a-50d, available to each evolutionary agent 30a-30d, and each evolutionary agent 30a-30d searches over a different, smaller space of the whole search space of planning decisions. The coevolutionary algorithms embodied in

coevolutionary agents 30a-30d have no direct means to search the full space of all planning decisions in the network architecture 10.

[0053]   While a single, centralized evolutionary agent compiles a list of all available decision resources at all nodes and explicitly searches the full space of planning decisions, such an operation can be slow and time-consuming in a distributed network environment. In contrast, the distributed co-evolutionary model of the invention allows each agent 30a-30d at each node 20a-20d to explore the full space of planning decisions using an information splicing operation in which information from each of the other nodes 20a-20d carried by mobile agents 60 is stochastically combined at the first node 20a-20d. The stochastic information splicing may be viewed as a crossover operation for combining information from the nodes 20a-20d.

[0054]   It is possible that as a practical matter, at some local nodes in a logical cluster of networked nodes, the evolutionary agents will not achieve convergence with the overall solution being produced by the other evolutionary agents as part of a global solution. This is inevitable to a distributed coevolutionary processing problem as some evolutionary agents will not have sufficient local information or useful local information for solving the global problem. In such case, evolutionary algorithms in the evolutionary agents will eliminate designs produced from the non-converging nodes as unsuitable for further consideration, while the remaining nodes with

good local information and advantageous resources for solving the global problem will continue to evolve to produce a solution accessible at substantially any one of the nodes 20a-20d in the architecture 10.

[0055]   As an example of an evolutionary algorithm which can be adapted for use with the distributed computation of the invention, let χ be the decision space. Then, $x \in \chi$ is the variable vector, and $x = (x_1, x_2, x_3, \ldots, x_p)$ represents a partition of the vector into p blocks. At any node i, x_i is its primary variable set 32a-32d, while $\overline{x_i}$ is the secondary variable set 34a-34d. Given a feasible space χ and a variable distribution, the evolutionary agent at each node i performs a local evolutionary search in its primary subspace χ_i, and so χ is the product space $\chi = \prod^P_{i=1} \chi_i$. $(x^*_i|\overline{x_i}) = \arg \min[x_i \in \chi_i]$ ψ $(x_i|\overline{x_i})$ is the optimizer in the restricted space $(\cdot|\overline{x_i})$. The evolutionary search in the primary subspace of each node i utilizes proportional selection and stochastic variational operations.

[0056]   Each evolutionary search described above is initialized with a randomly selected complete vector of variables x_g. Mobile agents facilitate the broadcast of this vector to all nodes 20a-20d in the network architecture 10. The evolutionary search starting from this point may be represented by the mapping $T_i : \chi \rightarrow \chi_i$ that generates the sequence:

$$x_{(i,g+m+1)} = Ti\left(x_{(1,g)}, \ldots x_{(i-1,g)}, x_{(i,g+m)}, x_{(i+1,g)}, \ldots, x_{(p,g)}\right), m \geq 0$$

[0057]   Then, as m increases:

$$x_g^{(i)} = (x_{(1,g)}, \ldots x_{(i-1,g)}, \; x_{(i,g+m)}, \; x_{(i+1,g)}, \ldots, \; x_{(p,g)})$$

and $x_g^{(i)}$ converges to $(x^*_i|x_i)$, where $x_g^{(i)}$ is the result of m generations of evolutionary search at node *i*, starting from point x_g.

[0058]   Now, let $Z_g = \{x_g, x_g^{(1)}, \ldots, x_g^{(p)}\}$ be a set of local results and the vector x_g, and let S: $\chi \to \chi$ represent the computation that selects that vector from $Z_g - x_g$ which has the highest fitness and makes it the new iterate x_{g+1} only if its fitness is greater than that of x_g. Otherwise, $x_{g+1} = x_g$. The computation $x_{g+1} = S(x_g)$ represents a global iteration that encapsulates the combined m-step local search at each node and the intercommunication operation, or coordination, that facilitates selection and update of new iterates. From the architectural perspective, mobile software agents 60 facilitate the coordination by transferring necessary information between coevolutionary agents 30a-30d.

[0059]   There are presently six preferred distributed coordination schemes, each of which uses information splicing. The schemes are referred to as local, joint, pool, elite local, elite joint and elite pool.

[0060]   The implementation of information splicing takes *p* vectors of the same dimension and creates a vector such that each of its coordinates is a random selection from the set of *p* coordinates along the same dimension. To help describe the coordination schemes,

the following assumptions are made: 1) the network environment has p network nodes; 2) $x_g^{(i)}$ is the best vector from node i at generation g; 3) $\{x_g^{(i)}\}$ is a set of vectors from node i at generation g; 4) $\{x_g'\}$ is a set of randomly created vectors at generation g; and 5) y_g is the vector obtained by combining the best local result portions from each node.

[0061]   The six coordination schemes can then be described as follows.

[0062]   In the local coordination scheme, from the set $\{x_g^{(1)},\ldots,x_g^{(p)}, \{x_g'\}\}$, select the best one as the new global iterate. The set $\{x_g'\}$ consists of p elements created by splicing from the set $\{x_g^{(1)},\ldots,x_g^{(p)}\}$.

[0063]   The joint coordination scheme has the set $\{x_g^{(1)},\ldots,x_g^{(p)}, \{x_g'\}\} \cup y_g$ from which the best is selected as the new global iterate. The elements of set $\{x_g'\}$ are the same as in the local coordination scheme.

[0064]   For the pool coordination scheme, from the set $\{\{x_g^{(1)}\},\ldots,\{x_g^{(p)}\}, \{x_g'\}\}$ select the best as the new global iterate. Each set $\{x_g^{(i)}\}$ represents $t=5$ top performers from each node i, and the set $\{x_g'\}$ is created as described above for the local and joint schemes from a set of size (t x p) rather than a set of size p.

[0065]   For the elite local scheme, from the set $x_g \cup \{x_g^{(1)},\ldots,x_g^{(p)}, \{x_g'\}\}$, select the best as the new global iterate where x_g is the previous global iterate.

[0066]   In the elite joint scheme, select the best from

the set $X_g \cup \{x_g^{(1)}, \ldots, x_g^{(p)}, \{x_g'\}\} \cup y_g$ as the new global iterate.

[0067] And, in the elite pool scheme, select as the new global iterate the best from the set $X_g \cup \{\{x_g^{(1)}\}, \ldots, \{x_g^{(p)}\}, \{x_g'\}\}$.

[0068] One network system that can be used to implement the distributed co-evolutionary agent problem solving system uses JAVA programming language developed by Sun Microsystems Inc. The implementation executes over multiple processing units distributed over a network. The implementation is based on the use of the VOYAGER object request broker developed by ObjectSpace Inc. as the underlying distributed communications environment. The VOYAGER broker is described in the ObjectSpace VOYAGER ORB 3.3 Developer Guide (2000), incorporated herein in its entirety by reference. The VOYAGER program serves as a middle-ware layer that provides a location-transparent and standardized environment for execution of the JAVA modules. A significant advantage to using VOYAGER is that it simplifies the task of remote enabling applications modules by automatically adding this feature at run-time, and it supports the inter-node migration of modules. The latter feature is an important requirement for realizing the mobile agents 60 in architecture 10.

[0069] As will be readily apparent, there are many applications for the distributed coevolutionary problem solving architecture 10 of the invention. The following provide specific examples of how the distributed coevolutionary problem solving

architecture 10 can be used to rapidly provide solutions to complex problems.

[0070]   Planning new product designs by coordinating between designers, suppliers and manufacturers is a very complex problem which is dependent on many factors, including availability of parts and manufacturing resources, and costs for parts and tooling and assembly and the ability to generate efficient designs.

[0071]   Fig. 2 displays a pictorial model of the problem of integrated design, supplier and manufacturing planning for modular products where suppliers and manufacturing resources are network distributed. The mathematical structure of this planning task is given by the equation:

$$\min\{\psi(x): Ax = b, \ x \in Z_+^n\}$$

where x represents a complete decision vector, $\psi(\cdot)$ is a nonlinear objective function, A is a constraint matrix, and b is a constraint vector.

[0072]   A decision problem in this formulation consists of three assignment problems, A_1, A_2, and A_3, as represented by the corresponding arrows in Fig. 2. The assignment problem A_1 is the assignment of parts 210 from parts library 200 to one or more designs 510 in a pool 500 of possible designs. Assignment problem A2 is the assignment of suppliers 310 from a list of available suppliers 300 who can supply the parts 210 for a given design 510. Assignment problem A_3 is the

assignment of designs 510 to available manufacturers 410 in a manufacturing resource pool 400.

[0073]   As will be apparent, each of the assignments in each assignment problem A_1, A_2, A_3 contributes to the overall product cost and product realization time. Further, each assignment has a non-linear effect on the cost and time; that is, the effect cannot be evaluated as weighted sums. The assignment problem triple (A_1, A_2, A_3) constitutes a set of highly coupled problems and each of the assignments cannot be considered independent of the others. Product cost is computed as an aggregate of the cost of parts 210 in a given design 510 and the cost of manufacturing the design 510, while product realization time is computed as an aggregate of the cost of parts supply lead time and time to manufacture the design 510. The overall objective function that is to be minimized is an heuristic weighting of the product cost and an exponential function of the product realization time, as given by:

$$\psi(x) = C(x) e^{(T(x)-\alpha)/\beta}$$

where $C(x)$ and $T(x)$ respectively represent the product cost and product realization time for a complete design-supplier-manufacturing assignment x, and α and β are non-zero constants.

[0074]   Fig. 3 illustrates the organization of a networked environment 600 used to solve the problem depicted in Fig. 2 in the context of printed circuit board assemblies. The networked environment 600 of Fig. 3 is depicted as a high-level configuration that

consists of several logical clusters 700, 800, 900 of network nodes 720, 820 and 920 and a product design node 620. The nodes 720, 820, 920 in each logical cluster 700, 800, 900 correspond to a class of functionally equivalent resources, and typically are physically distributed across the entire network 600. In Fig. 3, the logical clusters 700, 800, 900 correspond to parts distributor nodes 720, printed circuit board fabricator nodes 920 and printed circuit assembly nodes 820.

[0075]   Each parts distributor nodes 720 in the parts distributor logical cluster 700 corresponds to a parts distributor or parts warehouse that stocks components parts from several manufacturers. Each node 920 in the printed circuit board (PCB) fabricator logical cluster 900 corresponds to a PCB manufacturer having one or more PCB manufacturing lines. Each printed circuit assembly node 820 of the printed circuit assembly logical cluster 800 corresponds to a manufacturing facility having alternative manufacturing lines, each of which is capable of manufacturing printed circuit assemblies given a design, or collection of parts, and an associated PCB to assemble the parts on.

[0076]   The product design node 620 generates functional specifications that serve as partial templates for virtual designs. While the search at a parts distributor node 720 is over the space of functionally equivalent designs and is achieved by selecting alternative parts and suppliers for those parts, the

search at a PCB fabricator node 920 is over the space of available board manufacturing resources, and the search at a printed circuit assembly node 820 is over the space of available assembly resources. Mobile agents 60 (not shown in Fig. 3) communicate results between the various nodes 620, 720, 820, 920, so that the final minimized result can be obtained from any of the nodes 620, 720, 820, 920.

[0077]   Another application of the distributed coevolutionary problem solving invention is in the context of an Internet or World Wide Web (WWW) search engine. Presently, the WWW as it is commonly known consists of a vast collection of diverse information which is estimated to be about 1.5 billion documents large and growing. A large percentage of that material is available in the form of Web pages whose content is organized according to a markup protocol, such as XML or HTML. Web pages frequently provide content-dependent links to other Web pages, and their organization may be visualized as a graph whose nodes are the Web pages, and whose edges, or connections between nodes, are the links between pages.

[0078]   Searching and organization of Web pages for rapid retrieval has been the critical focus of contemporary search engines, and without these search engines most of the information on the Web would be inaccessible to users. Known search engines are essentially user-queryable centralized databases which contain indexed maps of the information on the WWW. The indices in the databases are populated and refreshed on a

periodic basis by "crawlers" or "spiders" or "bots" that retrieve and parse Web pages by visiting nodes (pages) and following the edges (links) between nodes. Essentially, these crawlers employ one of many graph search techniques in an attempt to traverse, retrieve, and organize distributed content based on index terms.

[0079]   In addition to Web pages, there are also many searchable dynamic databases reached through individual Web pages which process directed queries posed at the entry Web page. Current crawlers are incapable of accessing and conducting searches on the content of these databases. The large size and dynamic qualities of these databases make it impractical for a crawler to index them, because it effectively requires replicating the database in the crawler search engine database, and constant change would quickly make the search engine database out of date. Further, most crawlers are not capable of making the structured, directed queries necessary to locate information in the dynamic databases.

[0080]   It is generally accepted that the WWW follows a widely distributed multi-database architecture. To a local user, any single database in the WWW environment appears as a centralized repository, while it appears as a distributed collection of databases to a global user who wants to access coupled content from several databases.

[0081]   The following describes the application of the coevolutionary problem solving method of the invention to a dynamic retrieval and globally optimal

organization, viewed from the perspective of search relevance, of logically interrelated information distributed across several WWW databases.

[0082]   First, assume there is a space of p database nodes available on the WWW. Let a query $q=(q_1, q_2,...,q_p)$ represent a partition and assignment of q over each of the p nodes. Let χ_i be the space of local results at node i due to sub-query q_i. As a consequence, χ is the product space of results $\chi = \prod^p_{i=1} \chi_i$. Let $x=(x_1, x_2,...,x_p) \in \chi$ represent a specific result.

[0083]   The planning problem is then written as:

$$\min\{\psi(x):x=(x_1, x_2,...,x_p), \; x_i \in \chi_i \forall_i$$

where $\psi(\cdot)$ is a metric that measures the search relevance of a global result. This problem can visualized as the search for an optimal space of joint results from a Cartesian space of result tuples, wherein optimality is measured with respect to the search relevance of global results.

[0084]   The organization of the networked environment for the database search application is naturally a collection of nodes over which the coevolutionary search process executes using the planning problem as a foundation. Coevolutionary agents are created with programming to evaluate the planning problem and distributed to each of the collection of nodes.

[0085]   Nevertheless, there is an advantage to consider a networked environment of logical node clusters (similar to that of Fig. 3), wherein each logical cluster represents a certain topic-based

specialization of available information. The role of the product design node 620 of Fig. 3, for example, in the search engine application would be the node at which the user is resident and generates the search queries.

[0086] The coevolutionary agents are created as a result of the user formulating search queries and local searches are performed by coevolutionary agents at each node 620, 720, 820, 920. Following the initial local searches based on the primary search variables and updating the evolutionary agent solutions with the local search results (the primary search variables), mobile agents are used to communicate the results of the local searches to the other coevolutionary agents resident at the other nodes 620, 720, 820, 920 in the system architecture. The coevolutionary agents are updated with the transported local search solutions (the secondary search variables for the agents at different nodes) from where the using one of the coordination schemes discussed above. The search and updating steps may be repeated to produce evolved solutions which are further optimized based on the underlying algorithm and are superior to those of prior generations.

[0087] Although the distributed coevolutionary problem solving method is discussed in terms of producing printed circuit boards and conducting database searches, clearly, the method is adaptable to solving other complex, coupled manufacturing or delivery problems or performing distributed database searches

across any collection of distributed sources.

[0088]   While a specific embodiment of the invention has been shown and described in detail to illustrate the application of the principles of the invention, it will be understood that the invention may be embodied otherwise without departing from such principles.